APPLICATION NOTE

JULY, 1991

USING THE SLAVE PORT "COMMUNICATION BETWEEN TWO CPUs"

STEVE McINTYRE
AMY O'DONNELL
INTEL CORPORATION

1.0 Communicating with other CPU Devices

1.1 What is the Slave Port All About?

Three devices in the MCS-96 microcontroller family have integrated a new peripheral which offers an alternative. solution to communication between two CPU devices. This peripheral, called the Slave Port, provides a means to implement a master/slave bus configuration (similar to that of a CPU <--> MEMORY hookup). These MCS-96 devices would ad as the MEMORY device in the CPU <--> MEMORY configuration.

Communication between two miaocontrollers is useful when functions cannot be carried out within a single microcontroller. Different methods of communications are available to the design engineer. A serial link between the two microcontrollers is the most common The advantages of this method include: only utilizing 2 pins from each device, no hardware protocol, and allowing for error detection before data storage. However, serial links are rather slow and involves software overhead to differentiate the data, addresses, and commands.

To increase the speed of communications, a parallel bus can be used. This requires more pins and a rather involved hardware and software protocol. The ideal parallel solution is using a DPRAM to hold the shared data information. This hardware solution is shown in figure 1.0. Using a DPRAM offers the easiest software flexibility between master and slave devices, but the hardware interconnect is using a demultiplexed bus. This requires even more pins than a simple parallel connection The DPRAM is also very costly and data error detection can be messy.

Consider a forth alternative with the advantage of all three previous solutions, without a wide set of drawbacks. The Slave Port on the 87C196KJ, 87C196KR, and 87C196KT is designed to bring the DPRAM off chip, inside the KX. Now the External Processor A (See figure 1.0) can simply read/write to the on-chip memory of the "slave" KJ/KR or KT device (Processor B).

The number of interconnects is not that of a serial connection, but it is less than a DPRAM solution. There is no hardware protocol (the slave port can interface with both multiplexed or de-multiplexed buses). The External Master Processor A simply reads or writes as if there were a DPRAM device on the bus. The bus cycle of either processor is not impacted (0 wait states), back to back reads/writes take about 5uS. Data error detection can also be maintained through the KJ/KR or KT software.

Place Figure 1.0 (DPRAM and KJ SLAVE port concept diagram).

2.0 Slave Port Overview

The Slave port is really a simple bus configuration that can interface to an external processor via an 8-bit address/data bus. The KJ/KR or KT device communicates with the external processor through the Slave Port registers. These register are either input only or output only. When viewed from the KJ/KR or KT side the SLPSTAT and P3REG are output only registers that are latched out the Port 3 pins when the SLPCS# is low and the SLPRD# is also low. The SLPCMD and P3PIN are input only registers that are written when the SLPCS# pin is low and the SLPWR# is also low.

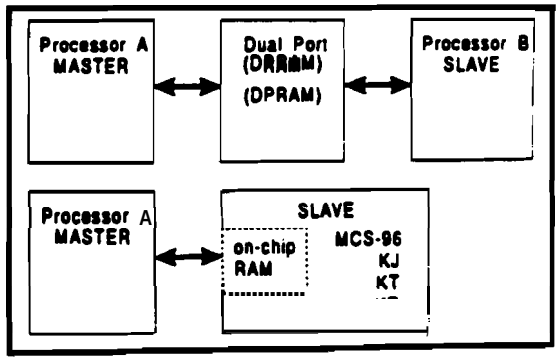

Figure 1-0. DPRAM and MCS-96 Slave Port

2.1 Slave Port Special Function Register Definition

When dealing with the Slave Port, the user MUST configure pins and speaal function register acccrding to his application. This section covers the configuration and initialization of these speaal function registers.

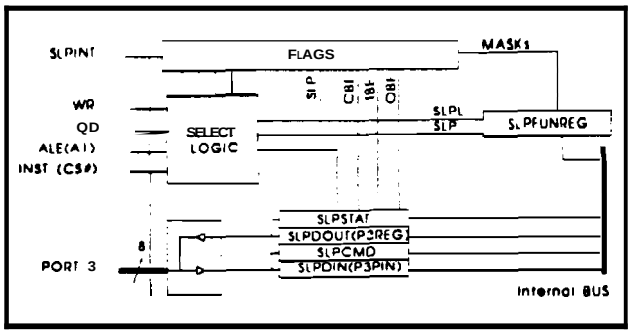

FIgure 2-0. Slave Port Block Diagram

2.1.1 Slave Port Special Function Registers:

There are 5 registers that the Slave Port communicates with: SLPFUNREG (location 1FFBh:byte), SLPSTAT (location 1FF8h:byte), SLPCMD (location 1FFAh:byte), P3REG (location 1FFCh:byte), and P3PIN (location 1FFEh:byte). All of which must be initialized before slave port functionality can be performed.

SLPFUNREG 1FFBh : byte

7	6	5	4	3	2	1	0
RSV	RSV	RSV	SME	SLP	SLPL	IBEmask	OBFmask

RSV: These bits should be written with zero's.

SME	This bit enables the slave port tor the PTS Shared Memory Mode (enable = "1").
SLP	"1" enables Stave Port
SLPL	"0" ALE is A1, "1" ALE is ALE
IBEmask	Enables IBE to effect SLPINT output pin
OBFmask	Enables OBF to effect SLPINT output pin

Figure 2-1. SLPFUNREG Register

SLP:

The SLP bit is used to enable the Slave Port functionality. When this bit is a "1" it will enable the Slave Port. Prior to setting (enabling) this bit make sure that the Port 3 and 5 pins are configured according to section 2.2. The Slave Port can be used with the PTS, or used through normal interrupts provided that the interrupts are enabled through the EI, EPTS, and INT_MASK, INT_MASK1 registers.

SME:

The SME (Shared memory Enable) is used to enable the "shared memory mode" of the slave port. (NOTE: this feature is not on the 87C196KR device, write this bit to a "0"). When this bit is a "0" the slave port works in the "Regular Slave Mode" of operation. (See "Regular Slave Mode" or "Shared Memory Mode" for details).

SLPL:

This bit is only used for the "Regular Slave Mode" of the slave port. When this bit is set to a "1", the SLPALE input pin is used to determine which register pair (SLPSTAT/SLPCMD or P3REG/P3PIN) is being accessed by the master processor. Typically this function is used for de-multiplexed busses. Typically the SLPALE pin would be connected to an Address line. When this address line is "1" the SLPSTAT/SLPCMD registers will be accessed, when "0" the P3REG/P3PIN will be accessed. This bit has no meaning in the "shared memory mode".

IBEmask:

This bit (bit 1) is used to determine if the "Input Buffer Empty" flag in the SLPSTAT register is used to toggle the SLPINT pins' output. (see figure 3.1 for block diagram of SLPINT structure). When this is a "1" the IBE bit in the SLPSTAT register will be masked with the SLPINT logic. When this bit is a " 0 any changes in the IBE bit of the SLPSTAT will have no affect on the SLPINT output pin. This bit has no meaning in "shared memory mode".

OBFmask:

This bit (bit 0) is used to determine if the "Output Buffer Full" flag in the SLPSTAT register is used to toggle the SLPINT pins' output. (see figure 3.1 for block diagram of SLPINT structure). When this is a "1" the OBF bit in the SLPSTAT register will be masked with the SLPINT logic. When this bit is a " 0 any changes in the OBF bit of the SLPSTAT will have no affect on the SLPINT output pin. This bit has non meaning in the "shared memory mode".

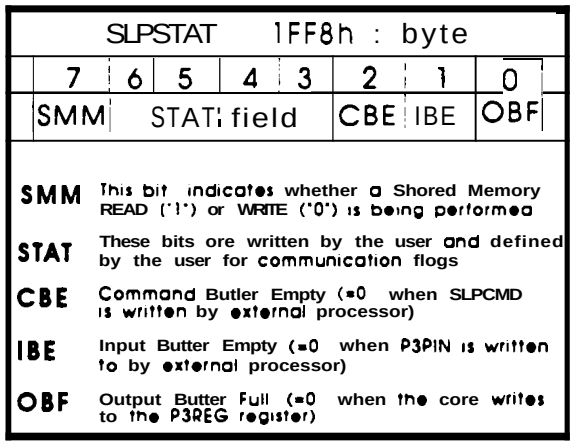

Figure 2-2. SLPSTAT Register

SMM:

This bit is a "read only" bit. It is used in the "shared memory mode" to indicate whether the bus interface logic received a write ("0) or a read ("1") operation. In the "Regular Slave Mode" this bit is part of the STAT field and can be read by the external master processor to **indicate/communicate** any user error codes, status, etc. See STAT field explanation

STAT:

These bits are intended to be used in the "Regular Slave Mode" by the slave and master devices to communicate certain error conditions. These bits are read and writable by the KX device software program. The external master processor can read these bits to determine the appropriate software corrective action.

CBE bit 2:

The "command buffer empty" bit is a status bit. This bit is cleared when the external processor writes to the SLPCMD **register,** and set to "1" when the KX device software or MS reads the SLPCMD **register**.

IBE bit 1:

The "Input Buffer Empty" bit is a status bit. This bit is cleared when the MPIN register is written by the external processor, and set to "1" when the KX device software or MS reads the P3PIN register. If the **IBEmask** in the SLPFUNREG is also set, KX software or PTS reading of the MPIN will toggle the SLPINT pin high.

OBF **bit0**:

The "Output Buffer Full" bit is a status bit. This bit is cleared when the external processor reads the MREG register, and set to "1" when the KX device software or PTS writes to the MREG register. If the **OBFmask** in the SLPFUNREG is also set, writing to the MREG by KX device software or PTS will toggle the SLPINT pin high.

2.2. **SLPCMD** register

The SLPCMD register is an 8-bit register that is written by the external processor. These eight bits of data can be used by the slave as communication protocol, or as an eight bit address pointing to internal or external memory that is being shared. In the "shared memory mode", this register is used to latch the 8 bit low address on the falling edge of SLPALE. In the examples that follow this register is only used as the low byte of the address being shared between master and slave processors.

2.3 P3PIN register

The P3PIN register is associated with the Port 3 pins. In non-slave port modes this register is used to read the logic level of the port 3 pins. In slave port functionality, this register is used for a similar function. The external processor writes data to this register and the slave KX device reads this register to receive that data. This register is also called the "input buffer". When the external processor writes to this register the IBE flag in the SLPSTAT register is written to a "0". When this register is read by the KX software program or PTS, the IBE flag in the SLPSTAT register is set to a "1". If the IBEmask in the SLPFUNREG is also set the SLPINT pin will be toggled to a "1" at the time of the KX device read.

2.4 P3REG register

The P3REG register is associated with the Port 3 pins. In non-slave port modes this register is used to latch the output logic levels to the outside world. In the slave port modes, this register has a similar function. The KX software or PTS will write data into this register while the external processor reads this output data. When the KX software or MS writes information to the P3REG register, the OBF flag in the SLPSTAT register will be set to a "1" and if the OBFmask is also set the SLPINT pin will be toggled high. When the external processor reads the P3REG register, this bit is cleared to a "0, indicating that the buffer is empty.

2.5 Other **Important Slave Port** Pins

There are three other pins in addition to the 8 pins of Port 3, SLPALE, and SLPINT. The SLPRD#, SLPWR#, and SLPCS#. Two are used as read and write signals (SLPRD# and SLPWR#). The rising edge of each signal is used to either write the data on the port 3 pins (P3REG/SLPSTAT) or latch the input data from port 3 into the P3PIN/SLPCMD register. All of this will not happen unless the "chip enable" pin (SLPCS#) is low during the entire bus cycle.

2.6 Configuring Ports 3 and 5 for use with the Slave Port

2.6.1 Port 5 Initialization

There are three registers that configure Port 5. Port 5 is the port that is shared with the system functions. These system functions (ALE, RD#, WR#, etc) are normally an output function. When configured for the Slave Port these pins are inputs. The only exception to this is the SLPINT pin; it is an output.

When configuring any port on the KJ/KR or KT devices the user program should first write the data to the PxREG register. Then, write the direction to the PxIO (on the KJ device this may be called the PxDIR register). Lastly, writing the mode to the PxSSEL (on the KJ device this may be called the PxMODE register).

For these devices the configuration code for Port 5 would look like this:

```
LDB     TEMP, #0FFH; write all ones to P5REG
STB     TEMP, P5REG[0]
LDB     TEMP, #0EFH; make all pins input but P5.4/SLPINT
STB     TEMP. P5IO[0]
LDB     TEMP, #10H; special select only the P5.4/SLPINT
STB     TEMP, P5SSEL[0]
```

2.6.2 Initializing Port 3

The KR c-stepping, KJ, and KT devices have a Port 3/4 Push-Push enable option on these ports. Configuration of the Port 3 pins should be set to Open-Drain. This is accomplished by simply clearing the MSB of the P34PPE register.

```
LDB     TEMP, P34PPE[0]  ; Read current state of PPE
ANDB    TEMP, #07FH      ; Clear the MSB bit
STB     TEMP. P34PPE[0]  ; Make Port 3 Open Drain.
```

2.6.3 Initializing the Slave Port Registers

Before any Slave Port activity can take place, the slave port SFRs must be initialized. The following code MUST be placed in the initialization code section after configuring Port 3 and 5. This code will select the mode (regular or shared memory) and setup the slave port to the application, and setup the SLPSTAT register in order to receive/send data over the Port 3 pins. The following is the recommended initialization code:

```
LDB     TEMP, #SLAVE_MODE
STB     TEMP, SLPFUNREG[0]  ; initialize the Slave Port
                            ;Modes
STB     RFF, P3REG[0]    ; write all 1's to Port 3 & set OBF
LDB     R00, SLPCMD[0]  ; clear command buffer register
LDB     R00, P3PIN[0]     ; clear input buffer register
LDB     TEMP, SLPSTAT[0]   ; read (CBE,IBE,OBF=111)
```

2.6.4 Slave Port and External Memory

The Slave Port uses the Port 3 and 5 pins that are normally used for accessing external program/data memory. If a system is designed that requires both a Slave Port function and External Memory, both functions can be supported through the use of the HOLD/HOLDA bus function.

The master device, prior to accessing the slaves' slave port will first insert a HOLD request. The slave device will acknowledge that the master may have the bus by the HLDA# signal. Then (and only then) may the master take control of the bus and read/write the slave port of the slave device.

The slave device must, upon receipt of the HOLD request, run from internal EPROM, or pause until the master has completed. The slave (if using interrupts) will return from the HOLD condition with the appropriate slave port interrupts pending.

NOTE: This feature is not recommended due to its complex nature and tricky timings of HOLD/HOLDA and slave port.

2.7 TOP 5 Issues with the Slave Port

1. Initialize Port 5 pins accordingly. P5.0/SLPALE, P5.1/SLPCS#, P5.2/SLPWR#, P5.3/SLPRD# setup as inputs and P5.4/SLPINT setup as Open-drain or Push-pull Output (if using this function).

2. Initialize Port 3 to Open-Drain, I/O.

3. Initialize the Slave Port according to the recommended initialization code in section 2.6.3

4. Before using interrupts or PTS with the Slave Port (CBF, IBF, OBE) the INT_MASK, INT_MASK1 must be enabled and the Interrupts/PTS must be enabled (EI / EPTS).

5. Use of the Slave Port with External Memory on Port 3/5 can be accomplished with the HOLD/HOLDA function. But, this is not recommended due to tricky timings and software.

3.0 Hardware Configuration

When using the slave port, some basic connection must be configured depending on the type of bus being connected. This hardware configuration (involving the slave port pins), is shown below for both a multiplexed and de-multiplexed bus.

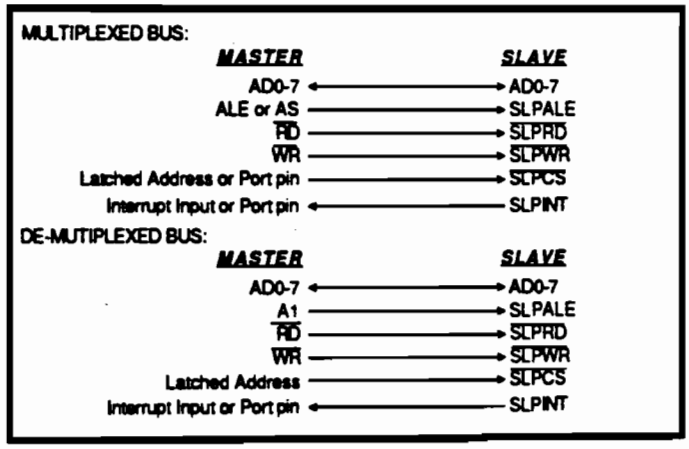

Figure 3-0 Master/Slave Inter-connections

When using a de-multiplexed bus, the regular slave mode must be used.

The configuration shown above allows the master to select the slave device by forcing the SLPCS# low. The master lets the slave know if a read or a write should be performed by either forcing the SLPRD# or SLPWR# pins low respectively. The data of a read or write data is latched on the rising edges of either one of these signals.

The slave communicates to the master when a read or write is complete through the SLPINT pin. The logic which operates this pin is shown below in Fig 3.1.

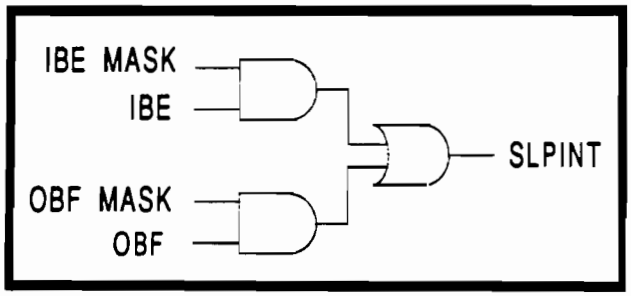

Figure 3-1. SLPINT Generation

The SLPINT pin is pulled low when the master writes to the P3PIN register or reads the P3REG. When the slave reads the P3PIN register, the IBE flag will be set. This causes the SLPINT pin to go high. This signals to the master device that the write is completed, and another write can be performed. A similar situation occurs with a read. When the slave device writes the data into the P3REG, the OBF flag is set, causing the SLPINT pin to go high. This tells the master that the data from the previous read cycle is now valid in the P3REG register. Notice that this read mechanism is a pipelined read. The address specified in the previous read cycle is fetched and placed in the P3REG register in order to be read by the master in the NEXT read cycle.

3.1 TOP 5 Issues Concerning Hardware Configuration:

1. The SLPINT pin cannot be driven low during a rising edge of RESET. If this happens a special Intel Test Mode (ONCE MODE) could be entered.

2. A de-multiplexed bus can ONLY be used in the regular slave port mode. Shared Memory Mode and Regular Slave Mode can use a multiplexed address/data bus as well.

3. The master reads the SLPSTAT/P3REG registers or writes the SLPCMD/P3PIN registers depending on the SLPALE/A1 state. ("0" = P3PIN/P3REG, "1" = SLPCMD/SLPSTAT)

4. The SLPINT pin will not be affected by the CBE flag.

5. The SLPINT pin goes low/high according to the following table.

	Regular Mode	Shared Memory Mode
Low (master):		
	Writes P3PIN	SLPCS = 0 and the falling
	Reads P3REG	edge of SLPALE
High (slave):		
	Reads P3PIN	Slave Reads P3PIN
	Writes P3REG	Slave Writes P3REG

4.0 Three Examples Using the Slave Port

4.1 Normal Slave Mode

4.1.1 Example Overview

The regular slave mode is available on the KR/KT or KJ microcontrollers. This is the only mode that can be directly used with a de-multiplexed bus. In this example, the master shares 256 bytes with the slave device. These 256 bytes may be located anywhere within the slave device memory by changing the base (MSB byte) address.

The low byte of the address is passed through the SLPCMD register. The base used in the example is 400h which locates the 256 bytes within the KX devices Code (400-4FFH).

The data to read will be located in the slaves' P3REG register, while the data written will be in the slaves' P3PIN register.

Address line one (A1) determines which pair of registers (P3PIN/P3REG or SLPCMD/SLPSTAT) in the slave device are accessed. When using a de-multiplexed bus, A1 is derived directly from the masters address output. The address output should be connected the slaves SLPALE pin (Refer to Fig 3.0b). For a multiplexed bus, the masters ALE pin is connected directly to the slaves SLPALE pin (Refer to Fig 3.0a). The A1 is derived from the AD1 value at the falling edge SLPALE and when SLPCS# is low.

For a de-multiplexed bus set the SLPL bit in the SLPFUNREG register to a zero. For a multiplexed bus the SLPL bit should be set to a one. The registers are accessed in the following manner:

When A1 = 0		
	the Master:	Writes to the P3PIN register or Reads the P3REG register
When A1 = 1		
	the Master:	Writes to the SLPCMD register or Reads the SLPSTAT register

4.1.2 Master Device Program

The master device has external memory locations that are dedicated for slave port accessing. These locations are somewhat arbitrary. The only restrictions is that when communicating with the SLPCMD/SLPSTAT the A1 must be a 1, and when communicating with the P3PIN/P3REG the A1 must be a 0.

A small master code segment illustrates the easy slave processor accesses:

```
FOR A WRITE:

EXT_P3PIN       EQU     0FFFDH  (A1=0)
EXT_SLPCMD      EQU     0FFFEH  (A1=1)

STB     DATA, EXT_P3PIN ; writes the data into the slaves
                        ; P3PIN
STB     ADDR, EXT_SLPCMD; writes the LSB address into
                        ; slaves SLPCMD
        ; WAIT FOR THE SLPINT TO GO HIGH BEFORE
        ; PERFORMING ANOTHER READ OR WRITE
```

This code segment writes the data to the P3PIN register first. This will clear the IBE flag in the slaves SLPSTAT register. The slave, seeing that the IBE is cleared will perform a data write at the BASE+SLPCMD address.

Performing a read cycle on the master is just as simple. The following code segment is used to illustrate how the master device would communicate that a read of a byte of data is needed and read that data from the slave processor:

```
FOR A READ:

EXT_P3REG       EQU     0FFFCH  (A1=0)
EXT_SLPCMD      EQU     0FFFEH  (A1=1)

LDB     TEMP, EXT_P3REG     ; clear slaves output buffer
STB     ADDR, EXT_SLPCMD    ;write LSB address into
                            ; SLPCMD
                ; Wait for SLPINT to go high
LDB     DATA, EXT_P3REG     ; read the data required
                            ; from the slave
```

This code segment reads the P3REG register first. This will make sure that the output buffer of the slave device is indeed empty. Next, it loads the address it wants to read in the SLPCMD register. This will cause an interrupt in the slave processor. The slave reads that location and stores it out the P3REG. At this time the SLPINT pins will go high telling the master that the data requested is ready to read. (An effective programmer could utilize this time doing other tasks). Data is then read from the P3REG register by the master.

4.1.3 Slave Device Program

After the initialization of the Slave Port and

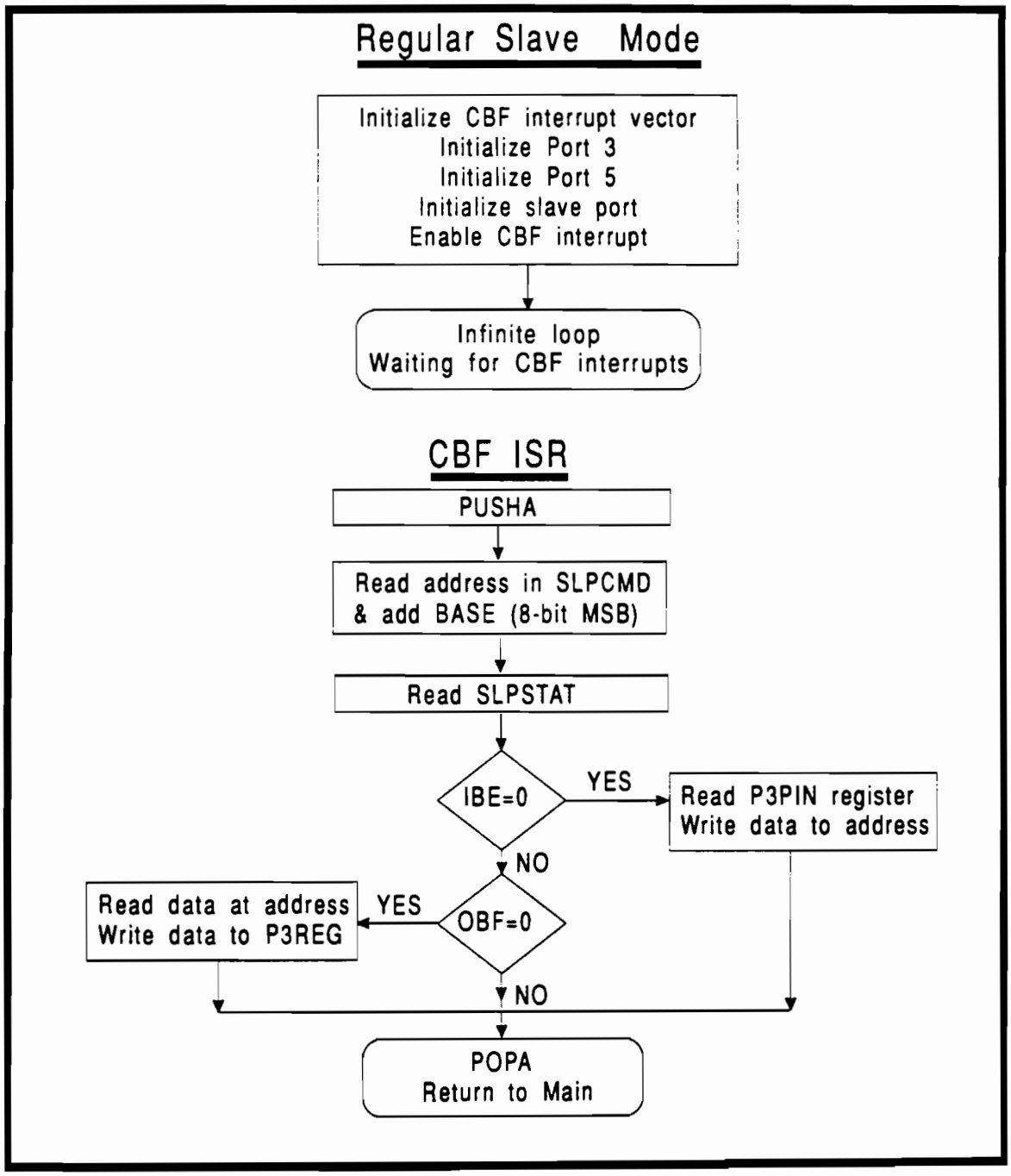

Figure 4-0. Regular Slave Mode

Port 3/5, the slave device program is strictly interrupt driven. When the slave device receives a byte in the SLPCMD register, the Command Buffer Full (CBF) interrupt occurs. In this interrupt service routine, the KX device determines whether the master device is requesting or sending data. This is determined through the OBF/IBE flags in the SLPSTAT register. Remembering that the master device cleared the output buffer before it loaded the SLPCMD register for a data read request and that the P3PIN register is written before the SLPCMD register is written for a write, only one flag will be cleared when the CBF interrupt is entered.

If the input buffer is full (IBE flag = 0) then a write of the data in the P3PIN register occurs at the address BASE + SLPCMD. Before a write takes place, the validity of the data or address can be checked at this point.

If the output buffer is empty, (OBF flag = 0) then the data located at the address BASE + SLPCMD is written to the P3REG so the master device can read it.

```
KJ_SLAVE   module   main
$include (sfr.kr)

       rseg   at     4ch

tempw:         dsw    1       ; Temp for use by monitor
       tw      equ    tempw:word
       tb      equ    tempw:byte
MailBox:       dsw    1       ; Pointer to Open Window over Slave Port
Base:          dsb    1       ; Base to be added to address

;
;     CHIP CONFIGURATION BYTE
;
       cseg   at     2018h
              dcw    20FDh  ; no limit rdy waits, and 8 bit mode
              dcw    20DEH  ; Watch Dog enabled
;
;     Interrupt Vectors
;
       cseg   at     200ch
              dcw    CMD_Full_ISR  ; Command Buffer Full SLP
;
;          BEGIN PROGRAM EXECUTION HERE
;
       cseg   at     2080h
reset_vector:
       di                           ; Disable interrupts
       dpts                         ; Disable PTS
       ld     sp,#200h                    ; Initialize stack pointer
       ClrB   WSR
;
;  Initialize Port 5
;
       ldb    tempw, #0ffh
       stb    tempw, P5REG[0]      ; p5reg = all 1's
       ldb    tempw, #0EFh
       stb    tempw, P5DIR[0]      ; p5io = P5.4 (SLPINT output)
       ldb    tempw, #10h
       stb    tempw, P5MODE[0]     ; p5ssel = SLPINT selected
;
;  Initialize Slave Port into normal mode
```

```
;
        ldb     tempw, #0Fh             ; SLP=1, SLPL=1, masks = "11"
        stb     tempw, SLPFUN[0]                ; store at SLPFUNREG
        stb     02h, P3REG[0]                   ; write to P3REG register
        ldb     R0, SLPCMD[0]           ; clear CMD buffer
        ldb     R0, P3PIN[0]                    ; clear P3PIN register
        ldb     R0, SLPSTAT[0]                  ; finish the initialization of SLP
;
; Now the SLP is ready
;
        ClrB    Int_pend
        ClrB    Int_pend1               ; make sure no pending interrupts are around
        LdB     Int_Mask,#40h                   ; Enable CBF interrupts
        LdB     base, #04h              ; The "Open" window is (0400-04FFh)
        EI                              ; Enable interrupts
monitor_pause:
        br      monitor_pause
;
; Command Buffer Full Normal Interrupt Service Routine
;
CMD_Full_ISR:
        Pusha
        ldbze   mailbox, SLPCMD[0]      ; read the SLPCMD value (mailbox=address)
        addb    mailbox+1, base                 ; window address is 400-4FFh
        ldb     tempw, SLPSTAT[0]       ; get SLPSTAT register

; if IBE=0 then master wants a write
        bbc     tempw, 1, Write_data

; if OBF=0 then master wants a read
        bbc     tempw, 0, Read_data

; if neither then RETURN ( An error must have occured because both bits
;       are set so a read or write cannot be performed)

Done_ISR:
        popa
        ret

Write_data:
        ldb     tempw, P3PIN[0]                 ; get data to write.
        stb     tempw, [mailbox]                ; write P3PIN @ SLPCMD+400h
        popa
        ret
Read_data:
        ldb     tempw, [mailbox]        ; get data to write to P3REG
        stb     tempw, P3REG[0]         ; write SLPCMD+400h data to P3REG
        popa
        ret
end
```

4.1.4 Memory Space Used and Timing Diagrams

The memory required for the master device would not be any more than that of using the DPRAM solution. However, the master processor performs two bus cycles for each byte written and three byte bus cycles for a read.

In the slave device, only 5 bytes are used: [Mailbox (word), Tempw (word), and Base (byte)] The time required to perform reads and writes in this mode using a 16 MHz clock is shown below:

READS: 91 states = 11.375 uS
WRITES: 86 states = 10.750 uS

These times do not incorporate the interrupt latency.

PUT IN TIMING DIAGRAM HERE

4.1.5 TOP 5 ISSUES Using the REGULAR SLAVE MODE:

1. Make sure the SME bit int the SLPFUNREG register equals "0" and the SLP bit = 1. The SLPL bit should be set according to the type of bus. (Multiplexed/"1" or De-Multiplexed/"0")

2. Make sure that the slave port initialization is done before any read or write operations.

3. The SLPCMD register only holds 8 bits of address. In order to access any/all 16 bit addresses, an offset can be added within the slaves Interrupt Service Routine.

4. When A1 = 0, the P3PIN/P3REG register pair is being accessed and when A1 = 1, the SLPCMD/SLPSTAT register pair is being accessed.

5. Make sure the master device performs ONLY byte operations when communicating with the slave device.

REGULAR SLAVE MODE TIMINGS

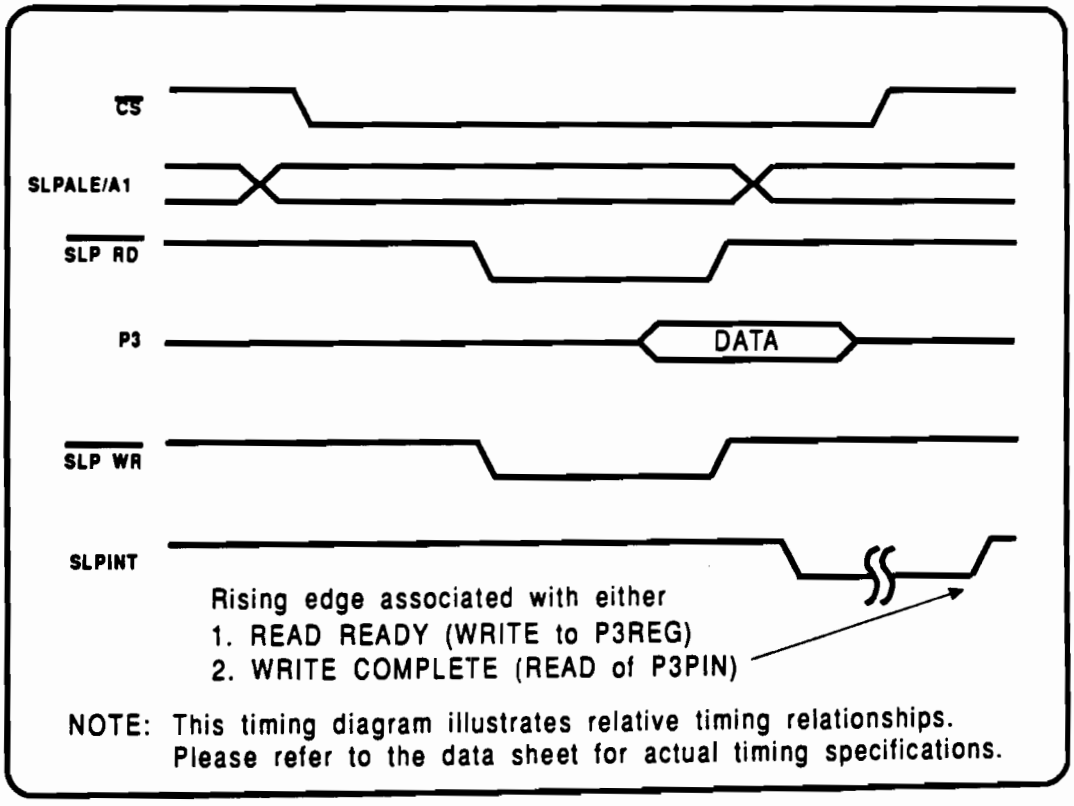

Figure 4-1. Regular Slave Mode Timings

4.2 Shared Memory Mode Using Normal Interrupts

4.2.1 Example Overview

This mode is available on the KT and KJ microcontrollers and a multiplexed bus must be used to operate the slave port in this mode. In this example, a 256 byte block of memory in the slave device is shared with the master. This 256 byte block may be located anywhere within the slaves memory map. For the code, the block is located in the KXs' code RAM. The high byte address is in the base address (BASE) and the low byte address is always in the SLPCMD register. The master utilizes the P3PIN register to store the data it wants to write. The P3REG is used to receive the data from a read.

The big difference between the shared memory mode using interrupts and the regular slave mode is the way the address is loaded into the SLPCMD register. The address (Low byte) is automatically loaded into the SLPCMD register on the falling edge of ALE. The data will be latched on the rising edge of read or write. This allows reads and writes to be done in one of the masters' bus cycle, instead of two or three in the other mode.

The time between the falling edge of ALE and the rising edge of RD# is not long enough to allow the slave processor to perform the read. Because of this, reads are pipelined in this mode as well. When the master requests a read, the data in that bus cycle is either "dummy" data or the data from the previous read. Even though reads are pipelined, writes can be performed in between reads with no corruption of the data waiting to be read. Knowing this the master can have write cycles have higher priority over reads.

4.2.2 Master Device Program

The master does not have to do very much to perform reads in this mode. It simply requests a read and receives data in one bus cycle from the previous read. Example code of how this is done is shown below.

```
OFFSET  EQU   #0FF00H

ADD  ADDR, #OFFSET  ; point to the ext addr
LDB  DATA, [ADDR]   ; read the slave device data
```

Remember that the DATA read is actual that of the previous read cycle. The ADDR driven will actually have the slave device perform an interrupt service routine to fetch the data at that address. The DATA at ADDR will be ready when the SLPINT pin goes high. (Rising Edge).

A write to the slave device is even simpler. The master must wait for the SLPINT pin to go high in between writes. Example code of how this is done is shown below.

```
OFFSET   EQU   #0FF00H

ADD  ADDR, #OFFSET   ; point to slave addr
STB  DATA, [ADDR]    ; store data at address
```

Write cycles are NOT pipelined. Due to this fact the write cycle can come between two reads and the pipelined read data is NOT corrupted.

The master still must wait for the SLPINT pin to go high between writes or reads. This time is only about 8 microseconds depending on the slave interrupt service routine.

4.2.3 Slave Device Program

The program included in this section displays how the slave device reacts to reads and writes requested by the master. The slave device knows that no matter what operation is selected the address will be latched into the command buffer (SLPCMD register). It takes advantage of this fact, and decides what operation needs to be performed by triggering off both the IBF and OBE interrupts.

When a IBF interrupt occurs, a write needs to be performed. The slave branches to that ISR and performs the write of the data in the P3PIN register at the address in the

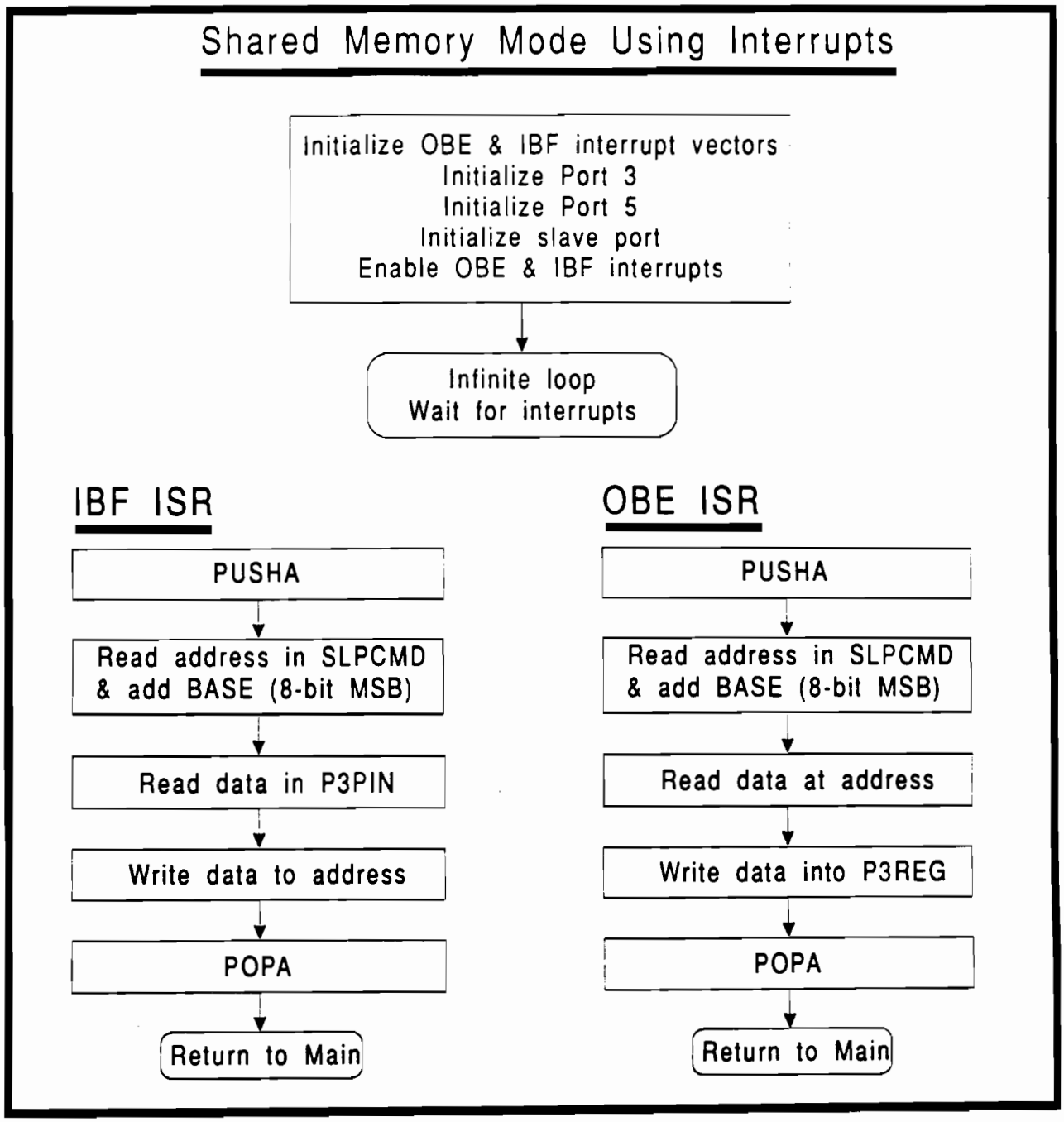

Figure 4-2. Shared Memory Mode Using Interrupts

SLPCMD register plus BASE. When the data is read from the P3PIN register, the SLPINT pin is toggled high letting the master device know another read/write may be performed.

When the slave gets an OBE interrupt, it knows that a read of the data at the address in the SLPCMD register needs to be performed. The slave branches to that ISR and loads the appropriate data into the P3REG. When the data is loaded into the P3REG register, the SLPINT pin is toggled high letting the master know another operation may be performed. Remember that the reads are pipelined.

```
KJ_SLAVE        module main

P5PIN           EQU     01FF7H:BYTE   ;
P5REG           EQU     01FF5H:BYTE   ;
P5DIR           EQU     01FF3H:BYTE   ;
P5MODE          EQU     01FF1H:BYTE   ;

P3PIN           EQU     01FFEH:BYTE   ;
P3REG           EQU     01FFCH:BYTE   ;
P4PIN           EQU     01FFFH:BYTE   ;
P4REG           EQU     01FFDH:BYTE   ;

SLPFUN          EQU     01FFBH:BYTE   ;
SLPCMD          EQU     01FFAH:BYTE   ;
SLPSTAT         EQU     01FF8H:BYTE   ;

R0              EQU     00H:WORD      ; R     ZERO REGISTER
PTSSEL          EQU     04H:WORD      ; R/W
PTSSRV          EQU     06H:WORD      ; R/W
INT_MASK        EQU     08H:BYTE      ; R/W
INT_PEND        EQU     09H:BYTE      ; R/W
INT_PEND1       EQU     12H:BYTE      ; R/W
INT_MASK1       EQU     13H:BYTE      ; R/W
WSR             EQU     14H:BYTE      ; R/W
SP              EQU     18H:WORD      ; R/W

rseg    at      4Ch

Addr:   dsw     1               ; Register used to calculate address
Temp:   dsb     1               ; Temporary storage register
Base:   dsb     1               ; Holds the 8 MSB of 16 bit address

;
;       INTERRUPT VECTORS
;
cseg    at      2008h
        dcw             OBE_ISR
;
cseg    at      200Ah
        dcw             IBF_ISR
;
;       CHIP CONFIGURATION BYTES
;
cseg    at      2018h
        dcw             20FDh           ; No limit rdy waits and 8 bit mode
        dcw             20DEh           ; Watch Dog enabled
;
;
;
```

```
;          BEGIN PROGRAM EXECUTION HERE
;
cseg   at     2080h
reset_vector:
       di                              ; Disable interrupts
       dpts                            ; Disable PTS routines
       ld      sp,#200h                ; Initialize stack pointer
       ClrB    WSR
;
;  Initialize Port 5
;
       ldb     temp, #0ffh
       stb     temp, P5REG[0]          ; p5reg = all 1's
       ldb     temp, #0EFh
       stb     temp, P5DIR[0]          ; p5io = P5.4 (SLPINT output)
       ldb     temp, #10h
       stb     temp, P5MODE[0]         ; p5ssel = SLPINT selected
;
;  Initialize Slave Port into normal mode
;
       ldb      temp, #1Ch             ; SME=1, SLP=1, SLPL=1, masks = "XX"
       stb      temp, SLPFUN[0]        ; store at SLPFUNREG
       stb      02h, P3REG[0]          ; write to P3REG register
       ldb      R0, SLPCMD[0]          ; clear CMD buffer
       ldb      R0, P3PIN[0]           ; clear P3PIN register
       ldb      R0, SLPSTAT[0]         ; finish the initialization of SLP
;
; Now the SLP is ready
;
       ClrB    Int_pend                ; Make sure no pending interrupts are around
       ClrB    Int_pend1
       LdB     Int_mask, #30h          ; Enable OBE and IBF interrupts
       LdB     Base, #04h              ; The "open" window is (0400-04FFh)
       EI                              ; Enable interrupts
pause:
       br      pause                   ; Wait for interrupts

IBF_ISR:
       Pusha                           ; Save flags
       LdBZE   Addr, SLPCMD[0]         ; Loads address from SLPCMD into Addr register
       AddB    Addr+1, Base            ; Adds a Base to the address (16 bit address)
       LdB     Temp, P3PIN[0]          ; Reads data from P3PIN and causes SLPINT to
                                       ; go high
       StB     Temp, [Addr]            ; Writes the data to the address
       Popa
       Ret

OBE_ISR:
       Pusha                           ; Save flags
       LdBZE   Addr, SLPCMD[0]         ; Loads address from SLPCMD into Addr register
       AddB    Addr+1, Base            ; Adds a base to the address
       LdB     Temp, [Addr]            ; Loads data from the address into a Temp reg
       StB     Temp, P3REG[0]          ; Writes the data to the P3REG and causes the
                                       ; SLPINT to go high
       Popa
       Ret

end
```

4.2.4 Memory Space and Timings

The memory space required for the sample code is equal to 4 bytes. [Addr (word), Temp (byte), Base (byte)] The time taken to perform reads and writes in this mode using a 16 MHz clock is shown below.

Read: 58 states = 7.25 microseconds
Write: 58 states = 7.25 microseconds

These times do not incorporate interrupt latency time or bus cycles used by the master device. Only one bus cycle from the master is needed to complete each read or write operation.

4.2.5 TOP 5 Issues of Shared Memory Mode Using Normal Interrupts

1. Initialize the Slave Port, Port 3 and Port 5 before using the slave port in the Shared Memory Mode.

2. Make sure that the SME bit and the SLP bit in the SLPFUNREG are both set to a "1".

3. The IBE and OBF Mask bits in the SLPFUNREG have no effect on the SLPINT pin operation.

4. Reads are pipelined. Each master bus cycle reads the previous addresses data.

5. Use a multiplexed bus with Shared Memory Mode.

SHARED MEMORY MODE TIMINGS

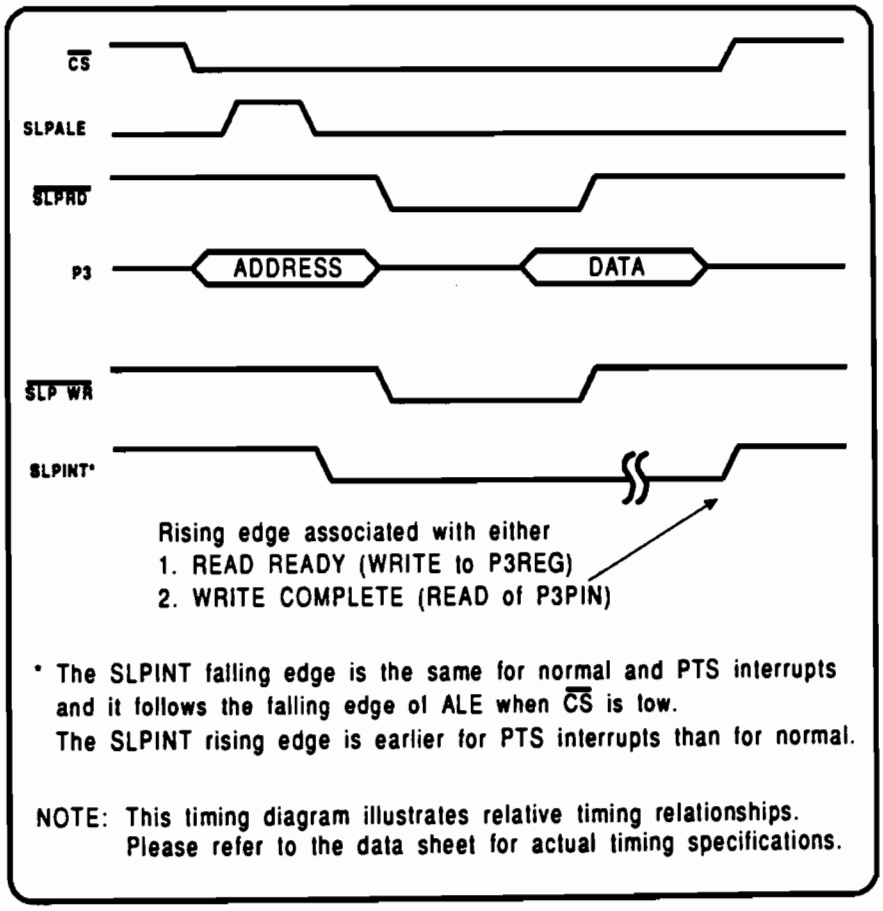

Figure 4-3. Shared Memory Mode Timings

4.3 Shared Memory Mode Using PTS Routines

4.3.1 Example Overview

In the previous example, the slave device interrupt service routines were very much the same. One performed reads from the SFR space to the memory block, and the other performed read from the memory block to the SFR space. The slave need only know which need to be performed in order to complete the Shared Memory Mode cycle. The PTS Shared Memory Mode is designed to perform this function.

This mode is only available on the KJ microcontroller and needs to have a multiplexed bus. In this example, a special PTS routine located in the KJ device is utilized to make reads and writes more automatic. The master shares a 256 byte block of memory located on the slave device. For the code given, the 256 byte memory block is located in the KJ's code RAM. The lower byte of address is latched into the SLPCMD register on the falling edge of ALE, while the upper byte of the address is kept in location BASE. The data will be loaded into the P3PIN register on writes and data from a read will be in the P3REG register. Reads are faster in this mode because of the PTS routine and are again pipelined.

4.3.2 Master Device Program

The master device program is identical to that in the previous example. Nothing has changed. The slave device only processes the information faster because of the PTS routine.

4.3.3 Setting up the PTS Control Block (Slave Program)

To make the data transactions automatic, a PTS Control Block must be set up within the slave device. A figure of the control block is shown in Fig 4.0.

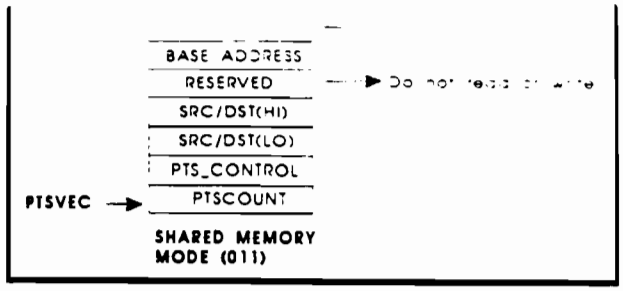

Figure 4-4. Shared Memory PTS MODE

The control block must be located within Register RAM and pointed to by the PTS interrupt vector for the IBF or OBE.

The PTS_COUNT determines the number of PTS cycles that will occur before a normal interrupt is taken. The maximum number allowed is 256. The PTS_CONTROL needs to be loaded with a 70h for Shared Memory Mode PTS. The SRC/DST (HI and LOW) need to point to the SLPSTAT register. The BASE register in the control block should contain the high byte of address being accessed. The reserved location should NOT be written, while Unused locations may be used as a scratchpad RAM by other routines.

4.3.4 Slave Device Program

The slave device treats reads and writes in the following manner. When either a IBF or OBE interrupt occurs, the slave branches to the PTS Control Block. The block looks at the SMM bit in the SLPSTAT register and decides if a read or write needs to be performed. If this bit = 1, a read takes place. If this bit = 0, a write takes place. The BASE address (High byte) and the address in the SLPCMD register (Low byte) are added together and the necessary operation (read/write) is performed using this address. The PTS_COUNT is decremented.

The SLPINT pin goes high when either the P3PIN register is read, or the P3REG register is written. This signals to the master that the bus cycle requested is complete and another read/write can take place.

If the PTS_COUNT is equal to a zero, the slave device vectors to a regular interrupt service routine which should reinitialize the

count and PTS interrupts.

A flow chart of the program and the code used to operate this mode is included below.

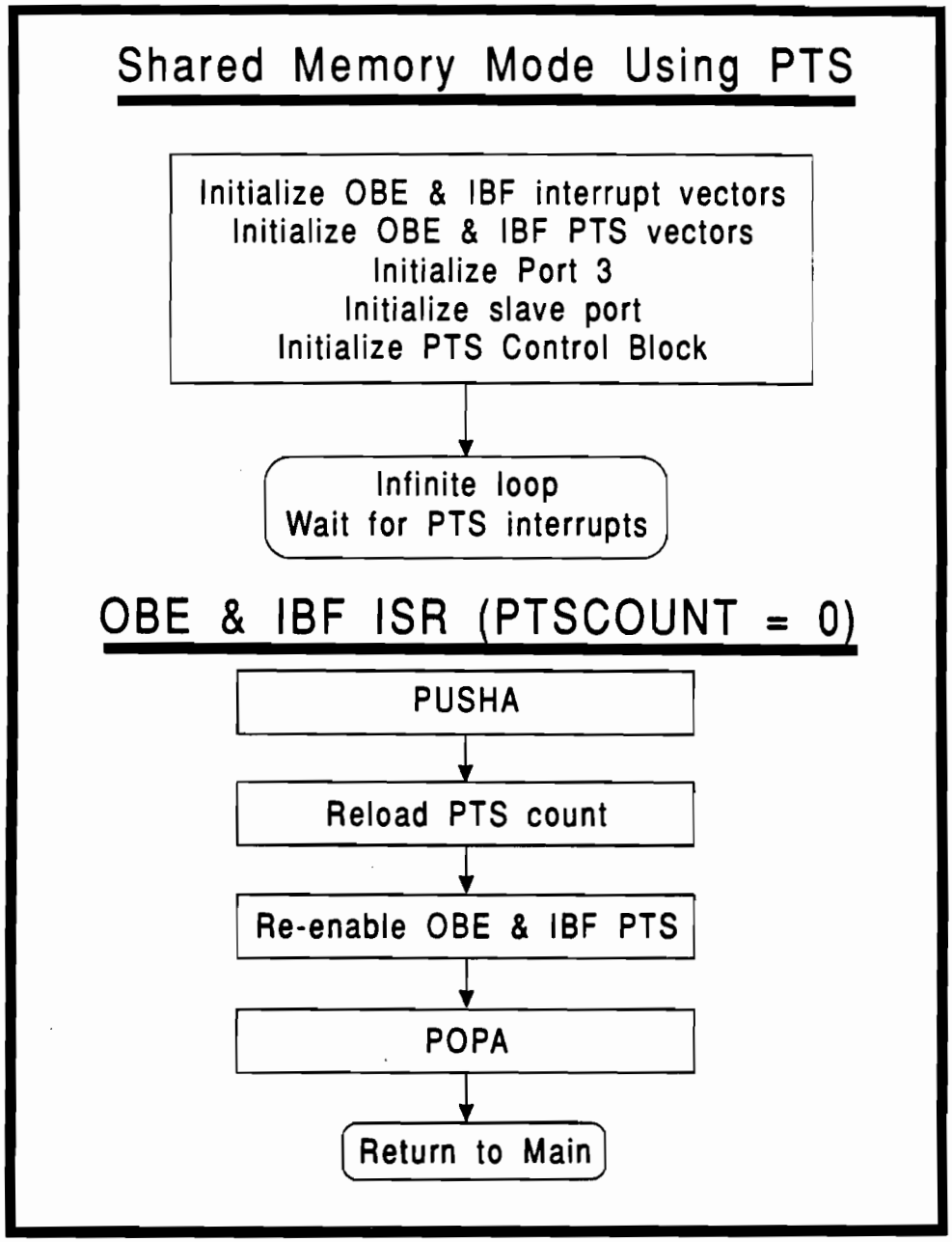

Figure 4-5. Shared Memory Mode Using PTS

```
KJ_SLAVE        module main

P5PIN           EQU     01FF7H:BYTE   ;
P5REG           EQU     01FF5H:BYTE   ;
P5DIR           EQU     01FF3H:BYTE   ;
P5MODE          EQU     01FF1H:BYTE   ;

P3PIN           EQU     01FFEH:BYTE   ;
P3REG           EQU     01FFCH:BYTE   ;
P4PIN           EQU     01FFFH:BYTE   ;
P4REG           EQU     01FFDH:BYTE   ;

SLPFUN          EQU     01FFBH:BYTE   ;
SLPCMD          EQU     01FFAH:BYTE   ;
SLPSTAT         EQU     01FF8H:BYTE   ;

R0              EQU     00H:WORD      ; R     ZERO REGISTER
PTSSEL          EQU     04H:WORD      ; R/W
PTSSRV          EQU     06H:WORD      ; R/W
INT_MASK        EQU     08H:BYTE      ; R/W
INT_PEND        EQU     09H:BYTE      ; R/W
INT_PEND1       EQU     12H:BYTE      ; R/W
INT_MASK1       EQU     13H:BYTE      ; R/W
WSR             EQU     14H:BYTE      ; R/W
SP              EQU     18H:WORD      ; R/W

        rseg    at      80h
Count:          dsb     1       ; Number of Xfers before Normal Interrupt
Cmd:            dsb     1       ; Command for Shared Memory Mode PTS
SLPPTR:         dsw     1       ; Pointer to SLPSTAT register (1FF8H)
Base:           dsb     1       ; 8 MSB bits of address "window"
open:           dsb     1       ; open byte to be used as scratch

;
;       Interrupt Vectors
;
        cseg    at      2008h
                dcw     OBE_IBF_ISR   ; Output Buffer Empty Interrupt
                dcw     OBE_IBF_ISR   ; Input Buffer Full Interrupt
        cseg    at      2048h
                dcw     Count         ; PTSCB for OBE interrupt
                dcw     Count         ; PTSCB for IBF interrupt
;
;    CHIP CONFIGURATION BYTE
;
        cseg    at      2018h
                dcw     20Fdh         ; no limit rdy waits, and 8 bit mode
                dcw     20DEH         ; Watch Dog enabled
;
;    BEGIN PROGRAM EXECUTION HERE
;
        cseg    at      2080h
reset_vector:
        di                            ; Disable interrupts
        dpts                          ; Disable PTS routines
        ld      sp,#200h              ; Initialize stack pointer
        ClrB    WSR
;
;   Initialize Port 5
;
        ldb     open, #0ffh
        stb     open, P5REG[0]        ; p5reg = all 1's
        ldb     open, #0EFh
        stb     open, P5DIR[0]        ; p5io = P5.4 (SLPINT output)
        ldb     open, #10h
        stb     open, P5MODE[0]       ; p5ssel = SLPINT selected
;
```

```
;  Initialize Slave Port into normal mode
;
        ldb     open, #1Ch              ; SME=1, SLP=1, SLPL=1, masks = "XX"
        stb     open, SLPFUN[0]         ; store at SLPFUNREG
        stb     02h, P3REG[0]           ; write to P3REG register
        ldb     R0, SLPCMD[0]           ; clear CMD buffer
        ldb     R0, P3PIN[0]            ; clear P3PIN register
        ldb     R0, SLPSTAT[0]          ; finish the initialization of SLP
;
; Now the SLP is ready
;
; Setup PTS control Block
;
        Ldb     Count, #0ffh            ; Count = max before normal interrupt
        Ldb     CMD, #01110000b         ; Command = 70H
        Ld      SLPPTR, #SLPSTAT        ; Pointer = SLPSTAT register
        Ldb     Base, #04h              ; Open Window = 0400 - 04FFh
;
        ClrB    Int_pend
        ClrB    Int_pend1               ; make sure no pending interrupts are around
        LdB     Int_Mask,#30h           ; Enable OBE IBF interrupts
        Ldb     WSR, #01h               ; Point to WSR = 1   (Bug in KJ)
        Ld      PTSSEL, #0030h          ; Enable PTS for OBE IBF interrupts
        ClrB    WSR                     ; Restore WSR
        EI
        EPTS
monitor_pause:
        br      monitor_pause           ; Wait for PTS cycles
;
; Only called when the PST_COUNT = 0
;
OBE_IBF_ISR:
        Pusha                           ; Save Flags
        LdB     Count, #0ffh            ; Reset the Count to Max (255)
        LdB     WSR, #01h               ; Set WSR=1 to get to PTSSEL (bug in KJ)
        Ld      PTSSEL, #0030h          ; Re - enable PTS for OBE IBF
        Popa
        ret
end
```

4.3.5 Memory Space Used and Timings

The memory space required for the Shared Memory Mode using PTS routines is equal to 6 bytes of register RAM. (PTS control block (PTS_COUNT:byte, PTS_CMD:byte, SRC/DST:word, BASE:byte, and the reserved location:byte). The time taken to perform the reads and writes using a 16MHz clock is as follows:

READS: 42 states = 5.250 microseconds
WRITES: 37 states = 4.625 microseconds

This is about 2 times the performance of the pervious example. This is a powerful tool for reads because it implements them fast. For writes it is fast but no error checking is being performed. The PTS could be used to do the reads, while the writes could be split off into a normal interrupt service routine. This would allow for fast reads (pipelined) through the PTS and quick writes through normal interrupts with error detection.

4.3.6 Top 5 Issues for Shared Memory Mode using PTS Routines

1. Initialize the Slave Port, Port 3, and Port 5 before and reads/writes take place.

2. Initialize the PTSSEL register and the INT_MASK registers to enable the IBF and OBE interrupts.

3. Initialize the PTS Control Block as specified and do not use the

Reserved location in this block. UNUSED locations can be used by other routines and is not used by the PTS.

4. The PTS Control Block MUST be located within Register RAM (0-1FFH) on a Quad Word boundary.

5. The master device must perform only byte reads/writes to the slave device and wait till the SLPINT output goes high before attempting additional bus cycles.